NASA Technical Memorandum 104234

A NASA/RAE Cooperation in the Development of a Real-Time Knowledge-Based Autopilot

Colin Daysh, Malcolm Corbin, Geoff Butler, Eugene L. Duke, Steven D. Belle, and Randal W. Brumbaugh

August 1991

National Aeronautics and Space Administration

NASA Technical Memorandum 104234

A NASA/RAE Cooperation in the Development of a Real-Time Knowledge-Based Autopilot

Colin Daysh, Malcolm Corbin, and Geoff Butler
Royal Aerospace Establishment, Farnborough, Hants, U.K.

Eugene L. Duke
Dryden Flight Research Facility, Edwards, CA

Steven D. Belle and Randal W. Brumbaugh
PRC Inc., Edwards, CA

1991

National Aeronautics and
Space Administration

Dryden Flight Research Facility
Edwards, California 93523-0273

CONTENTS

SUMMARY

As part of a United States–United Kingdom cooperative aeronautical research programme, a joint activity between the NASA Dryden Flight Research Facility and the Royal Aerospace Establishment on knowledge-based systems has been established. This joint activity is concerned with tools and techniques for the implementation and validation of real-time knowledge-based systems. This paper describes the proposed next stage of this research, in which some of the problems of implementing and validating a knowledge-based autopilot for a generic high-performance aircraft will be investigated.

NOMENCLATURE

AFSR	airworthiness and flight safety review
AI	artificial intelligence
a_n	normal acceleration
CADRE	cooperative advanced digital research experiment
CLIPS	'C' Language Production System
CRISP	cooperative real-time intelligent systems program(me)
FLEX	Fortran Library for EXpert System development
h	altitude
h_{com}	commanded altitude
Δh	altitude error
LISP	list processing language
KBS	knowledge-based system
KBAP	knowledge-based autopilot
RAE	Royal Aerospace Establishment
RAV	remotely augmented vehicle facility
V&V	verification and validation

INTRODUCTION

The research programme described in this paper will be the latest in a long-standing series of collaborations between the Dryden Flight Research Facility in the USA and the Royal Aerospace Establishment (RAE) in the UK. Previously, the cooperative advanced digital research experiment (CADRE) programme (ref. 1) was established in 1981 to investigate the applicability of nonlinear control techniques to a fly-by-wire aircraft. Nonlinear control laws developed at RAE were successfully flight tested on the NASA F-8 digital fly-by-wire aircraft using the remotely augmented vehicle (RAV) facility (ref. 2), in which ground-based computers are used to control a piloted aircraft remotely by way of telemetry links.

More recently, a collaborative project on knowledge-based systems (KBS) was undertaken (ref. 3) to investigate some of the real-time aspects of applying such techniques. Under this programme, a prototype flight status monitor for the X-29 research aircraft, which had been designed in a non-real-time form by NASA, was reimplemented at RAE in the Muse real-time KBS language (refs. 4, 5). This gave a significant speed improvement over the original and, though it still fell some way short of full

real-time operation, considerable insights were gained into the requirements which any real-time system would have to meet.

The proposed latest programme will go further in the investigation of real-time KBS design. Known informally as the cooperative real-time intelligent systems program(me) (CRISP), it will investigate the implementation and validation of a knowledge-based autopilot (KBAP). While it is recognised that this problem is probably more appropriately tackled by conventional algorithmic techniques, the KBAP provides a simple, well-defined yet real problem within which to explore, develop, and demonstrate real-time KBS concepts and validation and verification techniques for mission-critical systems.

A prototype autopilot has already been implemented using the NASA-developed KBS tool CLIPS, the C Language Production System (ref. 6). However, this implementation was shown to be fairly slow, and hence inappropriate for a real-time flight-control application. The RAE has developed the Fortran Library for EXpert System Development (FLEX, ref. 7), which has been demonstrated to be an order of magnitude faster than other KBS tools on benchmark problems (ref. 8). This paper discusses the likely best method of approaching the reimplementation of the CLIPS autopilot rules in FLEX to produce a KBAP which runs in real time. Some preliminary performance estimates are presented in the following paragraphs, based on work done using a generic rule set typical of the form likely to be taken by the final KBAP system.

One of the main areas of interest in this project is to establish a route by which a system based on KBS techniques could be validated as fit for flight. NASA Dryden engineers have considerable experience in the validation of more conventional algorithmic avionic systems (refs. 9–11) and are keen to investigate ways of extending these to cover the more diffuse behaviour exhibited by a KBS. If the work is successful, it is hoped that the KBAP would be flight tested in future phases of the project.

AN OVERVIEW OF THE KNOWLEDGE-BASED SYSTEM TOOL KITS

CLIPS Expert System Shell

CLIPS is an expert system shell with a list-processing language (LISP)-like syntax developed at the NASA Johnson Space Flight Centre. The basic elements of CLIPS are a fact list; comprising global memory for data, a knowledge base containing all of the rules, and an inference engine which controls overall execution.

A programme written in CLIPS consists of rules and facts. The inference engine decides which rules should be executed. Basically, rules fire (are executed) in much the same way that an IF THEN statement is executed in a procedural language such as Ada. That is, the rule fire IF certain conditions are true, and it THEN executes a set of actions. The conditions that fire the rule are facts. The rule only fires if certain facts have been asserted (i.e., the fact exists and is true).

CLIPS has variables available in which to store values. This is a very powerful tool which enhances the way in which rules and facts can be manipulated. Variables in CLIPS are written in the syntax of a question mark followed by a variable name. For example

?x
?value
?colour
?sound

One of the most useful, and most powerful applications of variables is in pattern matching. This is where the facts on the LHS of a rule are partially replaced by variables. For example the rule

```
(defrule variable-example
     (blue exterior)
     (?colour interior)
⇒
     (assert (interior-colour ?colour)))
```

will be fired by the "blue exterior," and by any fact which has two fields, with the word "interior" as its second field. It will then assert a fact that records the first field of the second fact. For example, the rule will be fired when the following facts are asserted

(blue exterior)
(pink interior)

The rule will assert the fact (interior-colour pink).

Another useful feature of CLIPS is its ability to do calculations within rules. Expressions to be calculated must be written in prefix form, (i.e., the expression 2 + 3 would be written (+2 3)). Again this is where CLIPS resembles LISP. This facility can be used to assert facts, for example

(assert (answer = (+ 2 3)))

would assert the fact "answer 5." It is possible to use variables within expressions, for example

(assert (answer =(- ?x ?y)))

This would calculate ?x - ?y, using the actual values assigned to the variables ?x and ?y, and then assert the fact to record the answer

Control within a CLIPS program can be achieved by using facts as controls but CLIPS provides a more direct method of control through salience. This allows assignment of priority to rules, to ensure that the highest priority rule in a set will fire first even if others are available to fire also.

CLIPS provides several commands to help in debugging. One command allows you to continuously watch facts being asserted and retracted. Whenever a fact is asserted or retracted, it will be displayed as such. Assertion of the fact (new_fact) would cause the display

⇒ f-1 (new_fact)

If this fact was then retracted, the display would be

⇐= f-2 (new_fact).

It is also possible to watch rules and activations on the agenda as the program is executing.

Fortran Library for Expert Systems

The FLEX is a library of Fortran 77 subroutines for developing expert system modules to interface directly with Fortran programs. It was developed at RAE (ref. 7) and has already found application in the field of aircraft structural design (ref. 12).

As well as the subroutines, a separate readable knowledge base is supported, together with forward, backward, and hypothesis–constraint inferencing. Basic explanation facilities are also provided. The FLEX library can be called from a higher level Fortran program to implement the inferencing procedures required by the problem. A knowledge base in FLEX consists of a number of separate rule bases contained within the knowledge base.

Rule Format

Rules are represented in the following form

Rule-identifier
IF property-a AND property-b.....
    THEN property-x AND property-t.....
    EXP=Explanations;

Disjunctive rules (i.e., rules whose antecedents contain facts linked by 'OR' functions) can also be used, in which case an 'AND' operator takes precedence over 'OR.' In other words,

'if A and B or C and D'

means if A and B are both true, or if C and D are both true.

The negative of a property is denoted by a 'not_' (optionally '-') directly before the name (e.g., -mammal means not a mammal).

REPRESENTATIONS OF FACTS

The facts which correspond to particular properties are stored in an array provided by the user. Each fact can be in one of three states; true, false, or unknown. There is no fact list as such, rather, when the rule bases are read in by the driving Fortran program, the properties and their associated values are stored in the aforementioned array. This is unlike a system such as CLIPS, where facts are only

stored in the fact list if they have been asserted. In FLEX, all the properties used in a rule base, as both antecedents or consequents, will be placed in the array when the rule base is read in. The fact value associated with each property can then be set explicitly. The setting of a fact value can be considered to be the same as asserting a fact in CLIPS, and similarly, setting a fact value to false can be considered the same as retracting a fact.

THE KNOWLEDGE-BASED AUTOPILOT

To illustrate the proposed approach to the verification and validation of KBSs, a rule-based longitudinal altitude-command autopilot example for a high-performance fighter aircraft has been designed. The example presented represents a single axis of a three-axis (longitudinal, lateral–directional, and velocity) controller. This controller is being developed and will be qualified as a mission-critical system as part of the research into validation methodologies for operation-critical KBSs.

A simplified representation of the aircraft and control system is shown in figure 1. The objective is to develop and to demonstrate a knowledge-based controller that produces command inputs to the aircraft control system based on a dynamic world model obtained from instruments on the aircraft and on a simple set of rules. While this task may not represent a suitable end application of a KBS (because it is easily performed by conventional algorithmic control laws), it provides a simple mission-critical application that is both easy to understand and easy to validate.

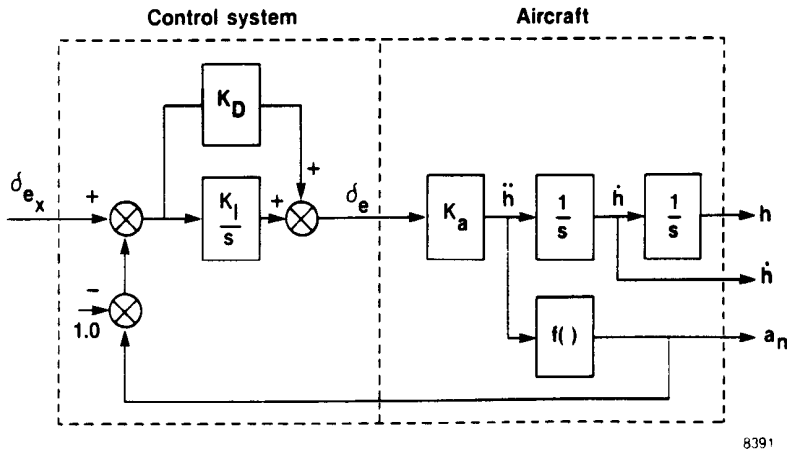

Figure 1. Simplified longitudinal model of the F-15 aircraft and its control system

The control task requires the autopilot system (whether based on conventional algorithms or a knowledge-based approach) to produce commands that cause the measured aircraft altitude (h) to be within some specified tolerance Δh of the commanded altitude h_{com}. Additionally, constraints are placed on the altitude rate and the normal acceleration a_n. The constraint on a_n is the same as a constraint on altitude acceleration, but an represents a more easily understood and easily measured physical quantity.

The initial requirement for this controller was that it control the aircraft in a consistent, repeatable manner at least as well as a pilot during both the transition mode (going from one altitude to another)

and the altitude-hold mode (controlling the aircraft about a specified altitude). The desire was to have it control the aircraft as well as a conventional algorithmic autopilot. An additional goal was to allow off-condition engagement so that the controller would also be effective even without benign initial (engagement) conditions.

These goals and requirements are similar to those initially imposed on the altitude-hold capabilities of the flight-test maneuver autopilot for the HiMAT vehicle (ref. 13). The constraints and tolerances were established as baseline figures. From this initial specification, a rule-based system was implemented that combined numeric and symbolic methods. This initial system was tested using a detailed nonlinear simulation model of the aircraft and its control system; the controller achieved excellent results for some initial conditions but performed poorly for many others. This initial result was typical of that experienced when evaluating the initial implementation of a conventional controller on a nonlinear simulation. After several iterations of this process, a fairly detailed statement of performance capabilities and limitations was established. This information, in essence, represents clarification of the statement of goals and requirements and serves as the basis of a functional specification for the system. An example of a prototype set of rules for the longitudinal altitude-hold section of the rule base is given in the following table.

Table 1. Preliminary rules for longitudinal altitude-hold autopilot.

Performance boundary rules	If altitude acceleration exceeds positive acceleration limit, move stick forward
	If altitude acceleration exceeds negative acceleration limit, move stick aft
	If predicted altitude rate exceeds positive altitude rate limit, trim stick forward
	If predicted altitude rate exceeds negative altitude rate limit, trim stick aft
Normal command rules	If altitude error is positive and predicted altitude rate is negative, trim stick aft
	If altitude error is negative and predicted altitude is positive, trim stick forward
	If predicted altitude error is positive and altitude error is small, click stick forward
	If predicted altitude error is negative and altitude error is small, click stick aft
	If predicted altitude error is positive and altitude error is large, trim stick forward
	If predicted altitude error is negative and altitude error is large, trim stick aft

Table 1. Concluded.

Definitions:	
move	large movement of stick
trim	intermediate movement of stick
click	small movement of stick

POSSIBLE IMPLEMENTATION USING FORTRAN LIBRARY FOR EXPERT SYSTEMS

The aim of this project is to reimplement an existing KBAP, supplied by NASA and written in CLIPS, in FLEX. The reasoning behind this is that FLEX has proven to be much faster than CLIPs and other currently available expert systems on benchmarking problems. It is hoped, therefore that a FLEX implementation of the KBAP can be made to run in real time.

The intention is to provide exactly the same functionality as the CLIPs version. Therefore the knowledge used in the CLIPS version has to be extracted and then rewrittten in a form that could be used by FLEX. Thus, the FLEX implementation would not require any knowledge engineering in the form on consulting a human expert on autopilots, since this work has already been done by NASA.

The usual method used for translating rules from one expert system language to another is to first rewrite the rules in English, using a structured format of IF THEN constructs. Once this knowledge has been extracted and rewritten into a FLEX knowledge base, Fortran driving code will have to be written to utilise it. The FLEX is not a self-contained expert system like CLIPs; rather it is a set of Fortran subroutines that utilise the decision making capabilities of expert systems. The difference between its capability and that of CLIPs will necessitate some changes to the division of labour between the algorithmic and knowledge-based parts of the system. In particular, FLEX does not allow arithmetic expressions within rules, so all calculations will have to be done by the algorithmic part of the autopilot which will assert facts for consideration by the rule base.

A preliminary investigation has been carried out on the conversion of a typical set of autopilot rules into FLEX. This has proved to give encouraging results. The combination of the restructuring as previously mentioned, and the greater efficiency of the FLEX inferencing procedure gave a speed improvement well in excess of a factor of ten over the CLIPs version, using a Sun 3 processor. This should provide ample scope for real-time operations, even if the final KBAP rules adopted prove to be more complex than this preliminary set.

Work on implementing KBSs in conventional languages is continuing at RAE, and an Ada-based KBS tool kit is likely to be available within the time scales of this project (ref. 14). It would be a valuable extension of the CRISP work to investigate a reimplementation using Ada as a comparison.

APPROACHES TO THE VALIDATION AND VERIFICATION OF THE KNOWLEDGE-BASED AUTOPILOT

The verification and validation (V&V) methodology used at Dryden is the same methodology that has actually been used for all flight-critical control systems in noncommercial aeronautical flight vehicles, including the F-18, Space Shuttle, and B-1 aircraft. This methodology uses a subset of the V&V techniques in use or advocated within the aeronautics community. The larger issues of certification and the validation of highly reliable, fault-tolerant systems have been of lesser concern than those of qualifying and conducting flight validation of flight-critical systems.

The basic methodology for the V&V of conventional operation-critical systems is directly applicable to the V&V of KBSs. In fact, if KBSs are to be used in operation-critical applications, the qualification of these KBSs will have to be performed within the context of established procedures and will have to address the requirements placed upon the qualification of conventional operation-critical systems.

The basis of the Dryden flight qualification and V&V methodology for embedded flight-critical systems is the incremental verification of systems components, integration testing, configuration management, and flight validation. The design specifications are transformed into hardware and software realizations. This transformation is not a straightforward, one-step process. The transformation of a design specification to an implemented prototype system requires the development and testing of numerous software procedures and hardware circuits, each of which is a prototype of some element in the larger system.

The implementation of system elements or components is supported by a variety of analysis tools and testing techniques (refs. 9, 10). The analysis tools used include failure modes and effects analysis, independent review, static verification, independent calculations, conjectures, and suspicions. This analysis is conducted on abstract models of the system or of the system components. Linear systems models, aggregate system models, block diagrams, schematics, source programs, specifications, and simulations are some of the main abstract models used. This analysis of abstract models is used to translate requirement and design specifications into a physical realization.

Simulation testing provides a closed-loop facility wherein the system in exposed to an environment that closely resembles the electronic and data environment in which the system must actually operate. Simulation also provides a facility for testing that the hardware and software of the system are integrated and operating together. Simulation is where the pilot (the system user) is first exposed to the system and allowed to evaluate it; the realism of the simulation is determined by the operating requirements for the flight application of the system.

The V&V methodology used for conventional, embedded operation-critical flight systems provides an established and accepted set of procedures upon which a methodology for KBSs can be based. While this position may be controversial in the AI community, the political and sociological realities of flight research and testing will ultimately dictate that any methodology for the validation of KBSs at least address the currently used methodology for conventional systems.

The proposed approach to the V&V of KBSs relies on the life cycle model shown in figure 2. The life cycle model for a KBS has been a topic of considerable concern to some who have addressed the validation of a KBS, and several models have been proposed (refs. 15–17). These models stress the development and prototyping process in a KBS. The motivation for developing these models is apparently to address the lack of a clear or well-defined statement of system goals and requirements and to highlight the prototyping process common in the development of KBSs. While the proponents of these models would probably contend that there is a fundamental difference between the life cycle of a KBS and a conventional system, another view is that this apparent difference is more reflective of the maturity of KBSs rather than of anything fundamental.

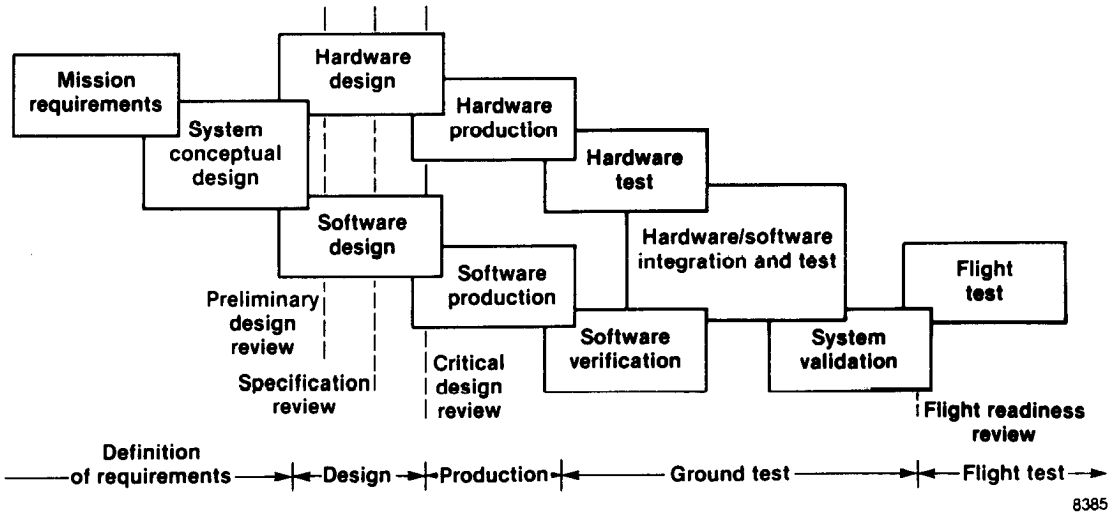

Figure 2. Dryden life cycle for research systems

Because KBSs are just emerging in operation-critical applications, there is little certainty of capabilities and limitations of these systems. The prototyping that is a common feature in the development of a KBS often represents an attempt to establish requirements for a given application. This definition of requirements, capabilities, and limitations through prototyping is not unlike that used in conventional systems when new techniques or applications are attempted. The difference is in the body of knowledge and experience behind the use of conventional systems as opposed to that of KBSs. Also reflected in this prototyping is the lack of maturity of artificial intelligence (AI) techniques in general that provides little basis for the selection of control and knowledge representation methods.

There are several issues that are almost certain to create problems for anyone attempting to validate operation-critical KBSs. Perhaps the most serious of these is an unwillingness to treat the current generation of KBSs out of the context of the promises of AI. The current generation of KBSs are not, in general, capable of learning or even modestly adaptive. These systems exhibit few nondeterministic properties. These KBSs may be complex but they are not unpredictable. But so long as there is this persistence in dwelling on the ultimate potential of AI systems instead of on the realities of the system being qualified, it is unlikely that an airworthiness and flight safety review (AFSR) panel would allow flight testing.

A further difficulty arises from the contention the KBSs do not always produce the correct answer. If this is true then a KBS can only be used for tasks in which their performance can be monitored and overridden by a human. Most operation-critical systems are required to perform without human intervention or with only high-level supervision or control. However, a KBS that does not always produce the optimum answer is acceptable as long as it never produces a wrong answer. This latter point is in fact one of the main V&V issues: operation-critical systems must be shown to produce acceptable solutions in all situations.

There are two key aspects of the proposed approach to the V&V of KBSs:

1. Development of a KBS to perform some task that is well known, well understood, and for which conventional V&V techniques are adequate, and

2. Incrementally and simultaneously expand both the KBS and the V&V techniques to more demanding and complex tasks.

The procedures used for verifying, qualifying, and validating conventional operation-critical flight systems at Dryden will be applied and modified as required. Because we ultimately plan to carry these experiments to flight using the rapid prototyping facility (ref. 2), this process will be performed under the sponsorship of and AFSR panel and will be the KBAP previously described that is being developed ultimately to perform aircraft maneuvers normally performed by highly trained pilots.

The research plan is to identify maneuvers of increasing difficulty and to build gradually more complex and adaptive KBS to accomplish those maneuvers. This will include prototyping, evaluation, and a series of initial operating capabilities that will evolve into a sequence of documented requirements for testing against each version of the system. This approach fits well within the model of and practice used with conventional digital systems.

CONCLUSIONS

This paper has described a proposed collaborative programme of research intended to approach the problem of the validation and verification of mission-critical knowledge-based systems in an incremental manner, using established procedures. The view presented in this paper is consistent with that proposed in Gault et al., in which, "A validation methodology for ultrahigh reliability, fault-tolerant systems must be based on a judicious combination of logical proofs, analytical modeling, and experimental testing."

This methodology must be supported by reliable validated development and test tools that lower the cost and reduce the schedule, if the goal of validation is to be achieved for either highly reliable, fault-tolerant systems or highly complex systems such as are envisioned for KBSs.

The work will focus on the real-time implementation of a knowledge-based autopilot, designed by NASA using a real-time knowledge-based system tool, FLEX, written at the Royal Aerospace Establishment. Preliminary results on a representative rule set indicate that FLEX will have more than sufficient speed for this application. The possibility also exists of an Ada implementation at a later stage.

The specific structures used within the real-time version may well influence some aspects of the approach taken towards validation, in particular the position of the boundary between the algorithmic and rule-based sections of the autopilot. It is hoped that, if the validation and verification work is successful, then flight tests, using the NASA Dryden Rapid Prototyping Facility, could be undertaken.

ACKNOWLEDGEMENT

The help of Mrs. M.A. Kirby in preparing this document is gratefully acknowledged.

REFERENCES

1. Butler, G.F., M.J. Corbin, S. Mepham, J.F. Stewart, and R.R. Larson, "NASA/RAE Collaboration on Nonlinear Control Using the F-8C Digital Fly-By-Wire Aircraft," 35th AGARD Guidance and Control Panel Symposium, Lisbon, Portugal, Paper 21, Oct. 1982.

2. Duke, E.L., R.W. Brumbaugh, and J.D. Disbrow, "A Rapid Prototyping Facility for Flight Research in Advance Systems Concepts," *Computer*, May 1989, pp. 61–66.

3. Butler, G.F. and E.L. Duke, "NASA/RAE Cooperation on a Knowledge Based Flight Status Monitor," AGARD 48th Guidance and Control Panel Symposium, Lisbon, Portugal, Paper 34, May 1989.

4. Reynolds, D., "MUSE: A Toolkit for Embedded Real-Time AI," *Blackboard Systems*, R. Englemore and T. Morgan, editors, Addison Wesley Publisher, 1988.

5. Miles, J.A.H., J.W. Daniel, and D.J. Mulvaney, "Real-Time Performance Comparison of a Knowledge-Based Data Fusion System using Muse, Art and Ada," Artificial Intelligence and Defence, AI'89, Avignon, France, May 1989, pp. 247-259.

6. Giarratano, J.C., "CLIPS User's Guide," Artificial Intelligence Section, Lyndon B. Johnson Space Center, June 1988.

7. Butler, G.F. and M.J. Corbin, "FLEX: Fortran Library for Expert Systems," RAE Working Paper MM 273/88, Dec. 1988.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1991	NASA Technical Memorandum

4. TITLE AND SUBTITLE

A NASA/RAE Cooperation in the Development of a Real-Time Knowledge-Based Autopilot

5. FUNDING NUMBERS

RTOP-505-66-71

6. AUTHOR(S)

Colin Daysh, Malcolm Corbin, Geoff Butler, Eugene L. Duke, Steven D. Belle, and Randal W. Brumbaugh

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

NASA Dryden Flight Research Facility
P.O. Box 273
Edwards, California 93523-0273

8. PERFORMING ORGANIZATION REPORT NUMBER

H-1727

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TM-104234

11. SUPPLEMENTARY NOTES

Prepared as a paper presented at the Avionics Panel Symposium in Lisbon, Portugal, May 1991, AGARD paper.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified — Unlimited
Subject Category 62

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

As part of a US/UK cooperative aeronautical research programme, a joint activity between the NASA Dryden Flight Research Facility and the Royal Aerospace Establishment on knowledge-based systems has been established. This joint activity is concerned with tools and techniques for the implementation and validation of real-time knowledge-based systems. This paper describes the proposed next stage of this research, in which some of the problems of implementing and validating a knowledge-based autopilot for a generic high-performance aircraft will be investigated.

14. SUBJECT TERMS

Knowledge-based system; Knowledge-based autopilot; Cooperative real-time intelligent systems program(me)

15. NUMBER OF PAGES

16. PRICE CODE
A0

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited